FILE NO. PAVI-26,402

METHOD AND APPARATUS FOR OPERATING A NEURAL NETWORK WITH MISSING AND/OR INCOMPLETE DATA

Inventor(s):

James D. Keeler, Eric J. Hartman and Ralph B. Ferguson

PAVI-26,402 PATENT
EXPRESS MAIL NO.: EL744748992US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

METHOD AND APPARATUS FOR OPERATING A NEURAL NETWORK WITH MISSING AND/OR INCOMPLETE DATA

TECHNICAL FIELD OF THE INVENTION

[0001] The present invention pertains in general to neural networks, and more particularly, to methods for estimating the accuracy of a trained neural network model, for determining the validity of the neural network's prediction, and for training neural networks having missing data in the input pattern and generating information as to the uncertainty in the data, this uncertainty utilized to control the output of the neural network.

CROSS-REFERENCE TO RELATED APPLICATIONS

[0002] This application is a continuation of U.S. Patent No. 6,591,254, to issue July 8, 2003, entitled "Method and Apparatus for Operating a Neural Network with Missing And/or Incomplete Data" (Atty. Dkt. No. PAVI-25,937), which is a continuation of U.S. Patent No. 6,314,414, issued November 6, 2001, entitled "Method For Operating a Neural Network With Missing and/or Incomplete Data" (Atty. Dkt. No. PAVI-24,542), which is a continuation of U.S. Patent No. 6,169,980 issued January 2, 2001, entitled "Method for Training and/or Testing a Neural Network on Missing and/or Incomplete Data" (Atty Dkt. No. PAVI-24,476), which is a continuation of U. S. Patent No. 5,819,006, issued October 6, 1998 (Atty. Dkt. No. PAVI-23,833), entitled "Method for Operating a Neural Network with Missing and/or Incomplete Data", which is a continuation of U.S. Patent No. 5,613,041, issued March 18, 1997, (Atty. Dkt. No. PAVI-20,965), entitled "Method and Apparatus for Operating a Neural Network With Missing and/or Incomplete Data."

BACKGROUND OF THE INVENTION

[0003] A common problem that is encountered in training neural networks for prediction, forecasting, pattern recognition, sensor validation and/or processing problems is that some of the training/testing patterns might be missing, corrupted, and/or incomplete. Prior systems merely discarded data with the result that some areas of the input space may not have been covered during training of the neural network. For example, if the network is utilized to learn the behavior of a chemical plant as a function of the historical sensor and control settings, these sensor readings are typically sampled electronically, entered by hand from gauge readings and/or entered by hand from laboratory results. It is a common occurrence that some or all of these readings may be missing at a given time. It is also common that the various values may be sampled on different time intervals. Additionally, any one value may be "bad" in the sense that after the value is entered, it may be determined by some method that a data item was, in fact, incorrect. Hence, if the data were plotted in a table, the result would be a partially filled-in table with intermittent missing data or "holes", these being reminiscent of the holes in Swiss cheese. These "holes" correspond to "bad" or "missing" data. The "Swiss-cheese" data table described above occurs quite often in real-world problems.

[0004] Conventional neural network training and testing methods require complete patterns such that they are required to discard patterns with missing or bad data. The deletion of the bad data in this manner is an inefficient method for training a neural network. For example, suppose that a neural network has ten inputs and ten outputs, and also suppose that one of the inputs or outputs happens to be missing at the desired time for fifty percent or more of the training patterns. Conventional methods would discard these patterns, leading to training for those patterns during the training mode and no reliable predicted output during the run mode. This is inefficient, considering that for this case more than ninety percent of the information is still there for the patterns that conventional methods would discard. The predicted output corresponding to those certain areas will be somewhat ambiguous and erroneous. In some situations, there may be as much as a 50% reduction in the overall data after screening bad or missing data. Additionally, experimental results have shown that neural network testing performance generally increases with more training data, such that throwing away bad or incomplete data decreases the overall performance of the neural network.

[0005] If a neural network is trained on a smaller amount of data, this decreases the overall confidence that one has in the predicted output. To date, no technique exists for predicting the integrity of the training operation of the network "on the fly" during the run mode. For each input data pattern in the input space, the neural network has a training integrity. If, for example, a large number of good data points existed during the training, a high confidence level would exist when the input data occurred in that region. However, if there were a region of the input space that was sparsely populated with good data, e.g., a large amount of bad data had been thrown out from there, the confidence level in the predicted output of a network would be very low. Although some prior techniques may exist for actually checking the actual training of the network, these techniques do not operate in a real-time run mode.

SUMMARY OF THE INVENTION

[0006] The present invention disclosed and claimed herein comprises a network for estimating the error in the prediction output space of a predictive system model for a prediction input space. The network includes an input for receiving an input vector comprising a plurality of input values that occupy the prediction input space. An output is operable to output an output prediction error vector that occupies an output space corresponding to the prediction output space of the system model. A processing layer maps the input space to the output space through a representation of the prediction error in the system model to provide said output prediction error vector.

[0007] In another aspect of the present invention, a data preprocessor is provided. The data preprocessor is operable to receive an unprocessed data input vector that is associated with substantially the same input space as the input vector. The unprocessed data input vector has associated therewith errors in certain portions of the input space. The preprocessor is operable to process the unprocessed data input vector to minimize the errors therein to provide the input vector on an output. The unprocessed data input in one embodiment is comprised of data having portions thereof that are unusable. The data preprocessor is operable to reconcile the unprocessed data to replace the unusable portion with reconciled data. Additionally, the data preprocessor is operable to output an uncertainty value for each value of the reconciled data that is output as the input vector.

[0008] In a further aspect of the present invention, the system model is comprised of a non-linear model having an input for receiving the input vector within the input space and an output for outputting a predicted output vector. A mapping function is provided that maps the input layer to the output layer for a non-linear model of a system. A control circuit is provided for controlling the prediction output vector such that a change can be effected therein in accordance with predetermined criteria.

A plurality of decision thresholds are provided that define predetermined threshold rates for the prediction error output. A decision processor is operable to compare the output prediction error vector with the decision thresholds and operate the output control to effect the predetermined changes whenever a predetermined relationship exists between the decision thresholds and the output prediction error vector.

[0009] In an even further aspect of the present invention, the non-linear representation of the system model is a trained representation that is trained on a finite set of input data within the input space. A validity model is provided that yields a representation of the validity of the predicted output of a system model for a given value in the input space. The validity model includes an input for receiving the input vector with an input space and an output for outputting a validity output vector corresponding to the output space. A processor is operable to generate the validity output vector in response to input of a predetermined value of the input vector and the location of the input vector within the input space. The value of the validity output vector corresponds to the relative amount of training data on which the system model was trained in the region of the input space about the value of the input vector.

[0010] In a yet further aspect of the present invention, the system model is trained by a predetermined training algorithm that utilizes a target output and a set of training data. During training, an uncertainty value is also received, representing the uncertainty of the input data. The training algorithm is modified during training as a function of the uncertainty value.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] For a more complete understanding of the present invention and the advantages thereof, reference is now made to the following description taken in conjunction with the accompanying Drawings in which:

FIGURE 1 illustrates an overall block diagram of the system model illustrating both a validity model and a prediction error model to process reconciled data and control the output with the use of the validity model and prediction error model;

FIGUREs 2a and 2c illustrates an overall block diagram of a method for training the system model utilizing the uncertainty generated during data reconciliation;

FIGURE 2b illustrates an example of reconciliation and the associated uncertainty;

FIGUREs 3a-3c illustrate data patterns representing the data distribution, the prediction error and the validity level;

FIGURE 4a illustrates a diagrammatic view of a data pattern sampled at two intervals illustrating a complete neural network pattern;

FIGURE 4b illustrates a diagrammatic view of a data pattern illustrating time merging of data;

FIGURE 5 illustrates an auto-encoding network for reconciling the input data to fill in bad or missing data;

FIGURE 6 illustrates a block diagram of the training operation for training the model;

FIGURE 7 illustrates an overall block diagram for training the validity model;

FIGUREs 8a and 8b illustrate examples of localized functions of the data for use with training the validity model;

FIGURE 9 illustrates a diagrammatic view of radial basis function centers in a two-dimensional space;

FIGURE 10 illustrates a diagrammatic view of the validity function;

FIGURE 11 illustrates distribution of training data and two test patterns for $\mathbf{x}_a$ and $\mathbf{x}_b$; and

FIGURE 12 illustrates an overall block diagram for generating the validity targets that are utilized during the training of the validity model.

DETAILED DESCRIPTION OF THE INVENTION

[0012] In FIGURE 1, there is illustrated an overall block diagram of the system of the present invention. A data input vector x(t) is provided that represents the input data occupying an input space. This data can have missing or bad data which must be replaced. This data replacement occurs in a data preprocess section 10, which is operable to reconcile the data patterns to fill in the bad or missing data and provide an output x'(t) vector. Additionally, the error or uncertainty vector $\mu_{x'}(t)$ is output. This represents the distribution of the data about the average reconciled data vector x'(t), and this is typically what is discarded in prior systems. The reconciled data x'(t) is input to a system model 12, which is realized with a neural network. The neural network is a conventional neural network that is comprised of an input layer for receiving the input vector and an output layer for providing a predicted output vector. The input layer is mapped to the output layer through a non-linear mapping function that is embodied in one or more hidden layers. This is a conventional type of architecture. As will be described hereinbelow, this network is trained through any one of a number of training algorithms and architectures such as Radial Basis Functions, Gaussian Bars, or conventional Backpropagation techniques. The Backpropagation learning technique is generally described in D.E. Rumelhart, G.E. Hinton & R.J. Williams, *Learning Internal Representations by Error Propagation* (in D.E. Rumelhart & J.L. McClennand, *Parallel Distributed Processing*, Chapter 8, Vol. 1, 1986), which document is incorporated herein by reference. However, Backpropagation techniques for training conventional neural networks is well known. The output of the system model 12 is a predicted output y(t). This is input to an output control circuit 14, which provides as an output a modified output vector y'(t). In general, whenever data is input to the system model 12, a predicted output results, the integrity thereof being a function of how well the network is trained.

[0013] In addition to the system model, a validity model 16 and a prediction- error model 18 are provided. The validity model 16 provides a model of the "validity" of the predicted output as a function of the "distribution" of data in the input space during the training operation. Any system model has given prediction errors associated therewith, which prediction errors are inherent in the architecture utilized. This assumes that the system model was trained with an adequate training data set. If not, then an additional source of error exists that is due to an inadequate distribution of training data at the location in the input space proximate to the input data. The validity model 16 provides a measure of this additional source of error. The prediction-error model 18 provides a model of the expected error of the predicted output.

[0014] A given system model has an associated prediction error which is a function of the architecture, which prediction error is premised upon an adequate set of training data over the entire input space. However, if there is an error or uncertainty associated with the set of training data, this error or uncertainty is additive to the inherent prediction error of the system model. The overall prediction error is distinguished from the validity in that validity is a function of the distribution of the training data over the input space and the prediction error is a function of the architecture of the system model and the associated error or uncertainty of the set of training data.

[0015] The output of the validity model 16 provides a validity output vector v(t), and the output of the prediction error model 18 provides an estimated prediction error vector e(t). These two output vectors are input to a decision processor 20, which output is used to generate a control signal for input to the output control 14. The decision processor 20 is operable to compare the output vectors v(t) and e(t) with the various decision thresholds which are input thereto from a decision threshold generator 22. Examples of the type of control that are provided are: if the accuracy is less than a control change recommendation, then no change is made. Otherwise, the controls are changed to the recommended value. Similarly, if the validity value is greater than the validity threshold, then the control recommendation is accepted. Otherwise, the control recommendation is not accepted. The output control 14 could also modify the predicted outputs. For example, in a control situation, an output control change value could be modified to result in only 50% of the change value for a given threshold, 25% of the change value for a second threshold and 0% of the change value for a third threshold.

[0016] Referring now to FIGURE 2a, there is illustrated one embodiment of a method for training the system model 12 utilizing the uncertainty $\mu(t)$ of the input training data. In general, learning of the system model 12 is achieved through any of a variety of neural network architectures, and algorithms such as Backpropagation, Radial Basis Functions or Gaussian Bars. The learning operation is adjusted such that a pattern with less data in the input space is trained with less importance. In the backpropagation technique, one method is to change the learning rate based on the uncertainty of a given pattern. The input uncertainty vector $\mu_{x'}(t)$ is input to an uncertainty training modifier 24, which provides control signals to the system model 12 during training.

[0017] The data pre-processor 10 calculates the data value x'(t) at the desired time "t" from other data values using a reconciliation technique such as linear estimate, spline-fit, box-car reconciliation or more elaborate techniques such as an auto-encoding neural network, described hereinbelow. All of these techniques are referred to as data reconciliation, with the input data x(t) reconciled with the output reconciled data x'(t). In general, x'(t) is a function of all of the raw values x(t) given at present and past times up to some maximum past time, Xmax. That is,

$$\begin{gathered} {}_1(t_N),\ x_2(t_N),\ \ldots\ x_n(t_N);\ x_1(t_{N-1}),x_1(t_{N-2}) \\ \ldots\ x_1(t_{N-1});\ x_1(t_1),\ x_2(t_1)\ \ldots\ x_n(t_1)) \end{gathered} \qquad \textbf{(001)}$$

where some of the values of $x_i(t_j)$ may be missing or bad.

[0018] This method of finding x'(t) using past values is strictly extrapolation. Since the system only has past values available during runtime mode, the values must be reconciled. The simplest method of doing this is to take the next extrapolated value $x'_i(t) = x_i(t_N)$; that is, take the last value that was reported. More elaborate extrapolation algorithms may use past values $x_i(t-\tau_{ij})$, $j \in t(o, \ldots i_{max})$. For example, linear extrapolation would use:

$$) = x_i(t_{N-1}) + \left[\frac{x_i(t_N) - x_i(t_{N-1})}{t_N - t_{N-1}} \right] t \quad ; \; t \tag{002}$$

Polynomial, spline-fit or neural-network extrapolation techniques use Equation 1. (See eg. W.H. Press, "Numerical Recipes", Cambridge University Press (1986), pp. 77-101) Training of the neural net would actually use interpolated values, i.e., Equation 2, wherein the case of interpolation $t_N > t$.

[0019] Any time values are extrapolated or interpolated, these values have some inherent uncertainty, $\mu_{x'}(t)$. The uncertainty may be given by *a priori* measurement or information and/or by the reconciliation technique. An estimate of the uncertainty $\mu_{x'}(t)$ in a reconciled value $x'(t)$ would be:

$$\vec{\mu}_{x'} = \begin{cases} \vec{\mu}_{ox'} + \mu_{1x'}t + \mu_{2x'}t^2 & \mu \leq \mu_{\max} \\ \vec{\mu}_{\max} & \mu \geq \mu_{\max} \end{cases} \tag{003}$$

where $\boldsymbol{\mu}_{\max}$ is the maximum uncertainty set as a parameter (such as the maximum range of data) and where:

$\mu_{ox'}$ is the a priori uncertainty

$$\mu_{1x'} = \left\langle \left| \frac{d\vec{x}'}{dt} \right| \right\rangle \tag{004}$$

i.e., the local velocity average magnitude and where:

$$\mu_{2x'} = \left\langle \left| \frac{1}{2} \frac{d^2\vec{x}'}{dt^2} \right| \right\rangle \qquad \textbf{(005)}$$

i.e., ½ the local acceleration average magnitude.

A plot of this is illustrated in FIGURE 2b.

[0020] Once the input uncertainty vector $\mu_{x'}(t)$ is determined, the missing or uncertain input values have to be treated differently than missing or uncertain output values. In this case, the error term backpropagated to each uncertain input is modified based on the input's uncertainty, whereas an error in the output affects the learning of all neuronal connections below that output. Since the uncertainty in the input is always reflected by a corresponding uncertainty in the output, this uncertainty in the output needs to be accounted for in the training of the system model 12, the overall uncertainty of the system, and the validity of the system's output.

[0021] The target output y(t) has the uncertainty thereof determined by a target preprocess block 26 which is substantially similar to the data preprocess block 10 in that it fills in bad or missing data. This generates a target input for input to a block 28, which comprises a layer that is linearly mapped to the output layer of the neural network in the system model 12. This provides the reconciled target y'(t).

[0022] Referring now to FIGURE 2c, there is illustrated an alternate specific embodiment wherein a system model 12 is trained on both the reconciled data x'(t) and the uncertainty $\mu_{x'}(t)$ in the reconciled data x'(t). This data is output from the data preprocess block 10 to a summation block 30 that is controlled on various passes through the model to either process the reconciled data x'(t) itself or to process the summation of the reconciled data x'(t) and the uncertainty $\mu_{x'}(t)$. Two outputs result, a predicted output p(t) and an uncertainty predicted output $\mu_p(t)$. These are input to a target error processor block 34, which also receives as inputs the reconciled target output y'(t) and the uncertainty in the reconciled target output $\mu_{y'}(t)$. This generates a value Δy_{total}. This value is utilized to calculate the modified Total Sum Squared (TSS) error function that is used for training the system model with either a Backpropagation Radial Basis Function or Gaussian Bar neural network.

[0023] In operation, a first forward pass is performed by controlling the summation block 30 to process only the reconciled data x'(t) to output the predicted output p(t). In a second pass, the sum of the reconciled data input x'(t) and the uncertainty input $\mu_{x'}(t)$ is provided as follows:

$$(t) + \vec{\mu}_{x'(t)} = (x_1' + \mu_{x_1'},\ x_2' + \mu_{x_2'},\ \ldots\ ,\ x_n' + \mu \qquad \textbf{(006)}$$

This results in the predicted output **p'(t)**. The predicted uncertainty **$\mu_p(t)$** is then calculated as follows:

$$\vec{p}'(t) - \vec{p}(t) = (p'_1 - p_1,\ p'_2 - p_2\ ,\ \ldots\ ,\ p'_m \qquad \textbf{(007)}$$

The total target error **Δy_{total}** is then set equal to the sum of the absolute values of **$\mu_p(t)$** and **$\mu_{y'}(t)$** as follows:

$$\Delta\vec{y}_{total} = (\ |\mu_{p_1}| + |\mu_{y_1'}|,\ |\mu_{p_2}| + |\mu_{y_2'}|\ ,\ \ldots) \qquad \textbf{(008)}$$

The output error function, the TSS error function, is then calculated with the modified uncertainty as follows:

$$E = \sum_{i=1}^{N_{PATS}} (\vec{y}_i - \vec{p}_i)^2 \left(1 - \frac{\Delta\vec{y}_{total_i}}{\Delta\vec{y}_{\max_i}}\right) \qquad \textbf{(009)}$$

where N_{PATS} is the number of training patterns.

For Backpropagation training, the weights W_{ij} are updated as follows:

$$W_{ij} = \begin{cases} -\eta \dfrac{\partial E}{\partial W_{ij}} & j \notin input \\ -\eta \dfrac{\partial E}{\partial W_{ij}} \left(1 - \left(\dfrac{|\mu_{x'_j}|^2}{|\mu_{x'_j \max}|^2}\right)\right)^2 & j \in input \end{cases} \qquad (010)$$

As such, the network can now have the weights thereof modified by an error function that accounts for uncertainty.

[0024] For neural networks that do not utilize Backpropagation, similar behavior can be achieved by training the system model through multiple passes through the same data set where random noise is added to the input patterns to simulate the effects of uncertainty in these patterns. In this training method, for each x'(t) and associated $\mu_{x'}(t)$, a random vector can be chosen by choosing each x''_i as $x''_i = x'_i + n_i$, wherein n_i is a noise term chosen from the distribution:

$$e^{(-x'^2_i \,/\, 2\mu_{x'_i}{}^2)} \qquad (011)$$

In this case:

$$\vec{\mu}_p(t) = f\,(\vec{x}') - f\,(\,\vec{x}'') \qquad (012)$$

Where f(**x(t)**) is the system model producing this system predicted output **p(t)**.

[0025] Referring now to FIGUREs 3a-3c, there are illustrated plots of the original training data, the system-model prediction and the prediction error, and the validity, respectively. In FIGURE 3a, the actual data input-target patterns are illustrated. It can be seen that the data varies in density and variance across the x-axis. Once the system model is trained, it yields a prediction, y (x), line 42. The system-model has an inherent prediction-error (due to inaccuracies in the training data). These prediction errors are illustrated by two dotted lines 44 and 46 that bound on either side of the predicted value on line 42. This represents basically the standard deviation of the data about the line 42. The validity is then determined, which is illustrated in FIGURE 3c. The validity is essentially a measure of the amount of training data at any point. It can be seen that the initial point of the curve has a high validity value, illustrated by reference numeral 48, and the latter part of the curve where a data was missing has a low level, as represented by reference numeral 50. Therefore, when one examines a neural network trained by the data in FIGURE 3a, one would expect the reliability or integrity of the neural network to be high as a function of the training data input thereto whenever a large amount of training data was present.

[0026] Referring now to FIGURE 4a, there is illustrated a data table with bad, missing, or incomplete data. The data table consists of data with time disposed along a vertical scale and the samples disposed along a horizontal scale. Each sample comprises many different pieces of data with two data intervals illustrated. It can be seen that when the data is examined for both the data sampled at the time interval 1 and the data sampled at the time interval 2, that some portions of the data result in incomplete patterns. This is illustrated by a dotted line 52, where it can be seen that some data is missing in the data sampled at time interval 1 and some is missing in time interval 2. A complete neural network pattern is illustrated box 54, where all the data is complete. Of interest is the time difference between the data sampled at time interval 1 and the data sampled at time interval 2. In time interval 1, the data is essentially present for all steps in time, whereas data sampled at time interval 2 is only sampled periodically relative to data sampled at time interval 1. As such, the reconciliation procedure fills in the missing data and also reconciles between the time samples in time interval 2 such that the data is complete for all time samples for both time interval 1 and time interval 2.

[0027] The neural network models that are utilized for time-series prediction and control require that the time-interval between successive training patterns be constant. Since the data that comes in from real-world systems is not always on the same time scale, it is desirable to time-merge the data before it can be used for training or running the neural network model. To achieve this time-merge operation, it may be necessary to extrapolate, interpolate, average or compress the data in each column over each time-region so as to give an input value x(t) that is on the appropriate time-scale. The reconciliation algorithm utilized may include linear estimates, spline-fits, boxcar algorithms, etc., or more elaborate techniques such as the auto-encoding network described hereinbelow. If the data is sampled too frequently in the time-interval, it will be necessary to smooth or average the data to get a sample on the desired time scale. This can be done by window averaging techniques, sparse-sample techniques or spline techniques.

[0028] Referring now to FIGURE 4b, there is illustrated an input data pattern and target output data pattern illustrating the pre-process operation for both preprocessing input data to provide time merged output data and also pre-processing the target output data to provide pre-processed target output data for training purposes. The data input x(t) is comprised of a vector with many inputs, $x_1(t)$, $x_2(t)$, ... $x_n(t)$, each of which can be on a different time scale. It is desirable that the output x'(t) be extrapolated or interpolated to insure that all data is present on a single time scale. For example, if the data at $x_1(t)$ were on a time scale of one sample every second, a sample represented by the time t_k, and the output time scale were desired to be the same, this would require time merging the rest of the data to that time scale. It can be seen that the data $x_2(t)$ occurs approximately once every three seconds, it also being noted that this may be asynchronous data, although it is illustrated as being synchronized. The data buffer in FIGURE 4b is illustrated in actual time. However, the data output as $x_1'(t)$ is reconciled with an uncertainty $\mu_{x'_1}(t)$ since the input time scale and the output time scale are the same, there will be no uncertainty. However, for the output $x'_2(t)$, the output will need to be reconciled and an uncertainty $\mu_{x'_2}(t)$ will exist. The reconciliation could be as simple as holding the last value of the input

$x_2(t)$ until a new value is input thereto, and then discarding the old value. In this manner, an output will always exist. This would also be the case for missing data. However, a reconciliation routine as described above could also be utilized to insure that data is always on the output for each time slice of the vector $x'(t)$. This also is the case with respect to the target output which is preprocessed to provide the preprocessed target output $y'(t)$.

[0029] Referring now to FIGURE 5, there is illustrated a diagrammatic view of an auto-encoding network utilized for the reconciliation operation. The network is comprised of an input layer of input nodes 60 and an output layer of output nodes 62. Three hidden layers 64, 66 and 68 are provided for mapping the layer 60 to the output layer 62 through a non-linear mapping algorithm. The input data patterns $x_1(t), x_2(t), \ldots, x_n(t)$ are input thereto, reconciled and reproduced over regions of missing data to provide the output data pattern $x_1'(t), x_2'(t), x_3'(t), \ldots, x_n'(t)$. This network can be trained via the backpropagation technique. Note that this system will reconcile the data over a given time base even if the data were not originally sampled over that time base such that data at two different sampling intervals can be synchronized in time.

[0030] The techniques described above involve primarily building, training and running a system model on data that may have missing parts, be on the wrong time-sale increment and/or possesses bad data points. The primary technique involves reconciliation over the bad or missing data and/or time-merging the data. However, once a model is built and trained, there are two other factors that should be taken into account before the model can be used to its full extent to solve a real-world problem. These two factors are the prediction accuracy of the model and the model validity. The model typically does not provide an accurate representation of the dynamics of the process that is modeled. Hence, the prediction output by the model will have some prediction-error $e(t)$ associated with each input pattern $x(t)$, where:

$$\vec{e}(t) = \vec{y}(t) - \vec{p}(t) \qquad \textbf{(013)}$$

This provides a difference between the actual output at time "t" and the predicted output at "t". The prediction error **e(t)** can be used to train a system that estimates the system-model accuracy. That is, a structure can be trained with an internal representation of the model prediction error **e(t)**. For most applications, predicting the magnitude ‖ **e(t)** ‖ of the error (rather than the direction) is sufficient. This prediction-error model is represented hereinbelow.

[0031] Referring now to FIGURE 6, there is illustrated a block diagram of the system for training the prediction-error model 18. The system of FIGURE 2c is utilized by first passing the reconciled input data x'(t) and the uncertainty $\mu_{x'}(t)$ through the trained system model 12, this training achieved in the process described with respect to FIGURE 2c. The target error Δy_{total} is calculated using the target error processor in accordance with the same process illustrated with respect to Equation 8, in addition to Δy as a function of "y". This is then input as a target to the prediction error model 18 with the inputs being the reconciled input data x'(t) and the uncertainty $\mu_{x'}(t)$. The prediction-error model can be instantiated in many ways, such as with a lookup table, or with a neural network. If instantiated as a neural network, it may be trained via conventional Backpropagation, Radial Basis functions, Gaussian Bars, or any other neural network training algorithm.

[0032] The measurement of the validity of a model is based primarily on the historical training data distribution. In general, neural networks are mathematical models that learn behavior from data. As such, they are only valid in the regions of data for which they were trained. Once they are trained and run in a feed-forward or test mode, (in a standard neural network) there is no way to distinguish, using the current state of the model lone, between a valid data point (a point in the region where the neural network was trained) versus an invalid data point (a point in a region where there was no data). To validate the integrity of the model prediction, a mechanism must be provided for keeping track of the model's valid regions.

[0033] Referring now to FIGURE 7, there is illustrated an overall block diagram of the processor for training the validity model 16. The data preprocess block 10 is utilized to provide the reconciled input data x'(t) to the input of the validity model 16. The input data x(t) and the reconciled input data x'(t) are input to a validity target generator 70 to generate the validity parameters for input to a layer 72.

A validity measure **v(x)** is defined as:

$$v(\vec{x}) = S\left(\sum_{i=1}^{N_{PATS}} a_i h_i \left(\vec{x}, \vec{x}_i\right) - b_i\right) \qquad \textbf{(014)}$$

where: **v(x)** is the validity of the point **x**

S is a saturating, monotonically increasing function such as a sigmoid:

$$S(z) = \frac{1}{1 + e^{-z}} \qquad \textbf{(015)}$$

a_i is a coefficient of importance, a free parameter,

h_i is a localized function of the data **x(t)** and the training data point **x_i(t)**,

N_{pats} is the total number of training patterns,

and

b_i is a bias parameter.

The parameter h_i is chosen to be a localized function of the data that is basically a function of the number of points in a local proximity to the point **x(t)**. As a specific embodiment, the following relationship for h_i is chosen:

$$\vec{x},\ \vec{x}_i) = \begin{cases} e^{-(\vec{x}-\vec{x}_i)^2/\sigma_i^2} & \|\vec{x}-\vec{x}_i\| < \\ o & \|\vec{x}-\vec{x}_i\| \geq \alpha \end{cases} \qquad \textbf{(016)}$$

The resultant function is illustrated in FIGURE 8a with the function cut of atασ so that far-away points do not contribute. Other functions such as the one illustrated in FIGURE 8b could also be used.

[0034] Referring now to FIGURE 9, there is illustrated an input space represented by inputs x_1 and x_2. It can be seen that there are three regions, each having centers x_1, x_2 and x_3, each having a given number of points n_1, n_2 and n_3, respectively, and a radius r_1, r_2 and r_3. The centers of the regions are defined by the clustering algorithms with the number of points determined therein.

[0035] Referring now to FIGURE 10, there is illustrated a representation of the validity function wherein the validity model 16 is illustrated as having the new data x(t) input thereto and the output v(x(t)) output therefrom. A dotted line is provided to the right of the validity model 16 illustrating the training mode wherein the inputs in the training mode are the historical data patterns $x_1(t)$, $x_2(t)$, ... $x_{Npats}(t)$, σ_i, α, a_i, b_i. In a specific embodiment, the values in the above are chosen such that $a_i = 1$, $b_i = 2$, for all i, $\sigma_i = 0.1$, $\alpha = 3$, for all i.

[0036] The Equation 14 can be difficult to compute, so it is more efficient to break the sum up into regions which are defined as follows:

$$\left\{ \sum_{i \varepsilon Cell_i} a_i h_i(\vec{x}, \vec{x}_i) + \sum_{j \varepsilon Cell_j} a_j h_j(\vec{x}, \vec{x}_j) + \right. \qquad \textbf{(017)}$$

where the cells are simple geometric divisions of the space, as illustrated in FIGURE 10, which depicts a test pattern.

[0037] In FIGURE 11, the test pattern $x_a(t)$ has a validity that is determined by cells C15, C16, C12 and C11 as long as the cell-size is greater than or equal to the cutoff $\alpha\sigma$, where the data point $x_b(t)$ is only influenced by cell C15 and C14. Hence, the algorithm for finding the validity is straightforward.

1) Train system model on training patterns $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots \mathbf{x}_{Npats})$
2) Train validity model by keeping track of $\mathbf{x}_1 \ldots \mathbf{x}_{Npats}$, e.g., via a binary tree or K-d tree.
3) Partition the data space into cells $C_1, C_2 \ldots C_{Ncells}$ (eg. K-d tree)
4) Determine which cell the new data point falls into, eg. cell-index $(\mathbf{x}) = (kx_1)(kx_2) \ldots (kx_n)$, if the cells are equally divided into k partitions/dimension and $x_i \varepsilon (0,1)$
5) Compute sum in cell
6) Compute sum in n-neighbors.
7) The validity function will then be defined as:

$$v(\vec{x}') = \sum_{Cell} \vec{x}' + f(d_i) \sum_{Neighbors} \qquad (018)$$

where d_i is the distance from $\mathbf{x}'$ to neighbor i, and $f(d_i)$ is a decreasing function of d_i.

Again, Equation 18 can be difficult to calculate. Furthermore, it may be the case that few data points fall into the individual cells. A useful approximation of the full sum may be made by including only those neighbors with large $f(d_i)$. A second, simpler, and faster way of computing the sums in Equation 18 is to approximate the sums by averaging all points in a region as follows:

$$') \cong S(N_1 a_1 h_1 (\vec{x}' , \vec{x}_1) + N_2 a_2 h_2 (\vec{x}' , \vec{x}_2) - \qquad (019)$$

$$v(\vec{x}') \cong S\left(\sum_{Regions} N_i a_i h_i (\vec{x}' , \vec{x}_i) - b \right) \qquad (020)$$

The region centers $\mathbf{x}_i$ can be selected as the centers of the cells x_i, or as the centers of k-d tree cells, or as the centers of Radial Basis functions that are selected via a k-means clustering algorithm.

[0038] Referring now to FIGURE 12, there is illustrated a block diagram of the validity model 16 for receiving the output of the pre-processor 10 and generating the validity value v(x'(t)). As described above, the output of the preprocessor 10 comprises both the reconciled data x'(t) and the uncertainty $\mu_{x'}(t)$. This is input to a region selector 76 which is operable to determine which region of the test pattern the reconciled data resides in. During training, a counter 78 is incremented to determine the number of points in the region over which the system model 12 was trained. This is stored on a region-by-region basis and, during a run mode, the incrementing operation that is controlled by a line 77 is disabled and only a region line 79 is activated to point to the region determined by the region selector 76. The output of the counter comprises the number of points in the region N_i, which is then input to a region activation block 80. The block 80 provides the function h(x'(t)), $x_i(t)$), which, as described above, is the localized function of the data x'(t) and the training data points $x'_i(t)$. The output of the region activation block 80 is input to a difference circuit 81 to subtract therefrom a validity bias value "b". This is essentially an offset correction which is an arbitrary number determined by the operator. The output of the difference circuit 81 is input to a sigmoidal function generator that provides the output v(x'(t)). The sigmoidal function provides a sigmoidal activation value for each output of the vector v(x'(t)).

[0039] In operation, the validity model 16 of FIGURE 12 allows for on-the-fly calculation of the validity estimation. This requires for the calculation the knowledge of the number of points in each region and knowledge of the region in which the input pattern resides. With this information, the estimation of the validity value can be determined. During the training mode, the increment line 77 is enabled such that the number of points in each region can be determined and stored in the counter 78. As described above, the run mode only requires output of the value N_i.

[0040] In the embodiment of FIGURE 7, the validity target generator 70 could utilize the structure of FIGURE 12 to calculate a target output for each value of x(t) input to the preprocessor 10. This would allow the validity model 16 to be realized with a neural network, which is then trained on the validity targets and the input data in accordance with a training algorithm such as backpropagation.

[0041] In summary, there has been provided a method for accounting for bad or missing data in an input data sequence utilized during the run mode of a neural network and in the training mode thereof. The bad or missing data is reconciled to provide a reconciled input data time series for input to the neural network that models the system. Additionally, the error that represents uncertainty of the predicted output as a function of the uncertainty of the data, or the manner in which the data behaves about a particular data point or region in the input space, is utilized to control the predicted system output. The uncertainty is modelled during the training phase in a neural network and this network utilized to provide a prediction of the uncertainty of the output. This can be utilized to control the output or modify the predicted system output value of the system model. Additionally, the relative amount of data that was present during training of the system is also utilized to provide a confidence value for the output. This validity model is operable to receive the reconciled data and the uncertainty to predict a validity value for the output of the system model. This is also used to control the output. Additionally, the uncertainty can be utilized to train the system model, such that in regions of high data uncertainty, a modification can be made to the network to modify the learning rate as a function of the desired output error during training. This output error is a function of the uncertainty of the predicted output.

[0042] Although the preferred embodiment has been described in detail, it should be understood that various changes, substitutions and alterations can be made therein without departing from the spirit and scope of the invention as defined by the appended claims.